

HƯỚNG DẪN CẤU HÌNH TRIỂN KHAI CI/CD LEVEL 3

I. Các thành phần hệ thống phục vụ triển khai CI/CD

STT	Tên hệ thống	Mô tả	Link ứng dụng
1	Jenkins	Hệ thống CI/CD thực hiện việc build/test tự động và triển khai mã nguồn lên môi trường dev, staging và môi trường production	10.60.156.96:8080
2	Gitlab	Hệ thống quản lý source code	10.60.156.11
3	Nexus	Hệ thống lưu trữ các sản phẩm phần mềm (artifact) và các thư viện được sử dụng trong quá trình build	10.60.156.26:8181 10.60.108.23:9001
4	Habor	Hệ thống quản lý images docker	10.60.156.72
5	Một số các công cụ khác		
	Ansible	Đối với các dự án chưa sử dụng K8s, ansible được sử dụng để deploy ứng dụng.	
	K8s	Sử dụng kubectl để deploy ứng dụng.	

II. Yêu cầu trước khi thực hiện cấu hình triển khai cần chuẩn bị một số thông tin sau:

1. Có tài khoản trên các hệ thống phục vụ triển khai CI/CD. Nếu chưa có tài khoản yêu cầu liên hệ đầu mối P.KTPM đ/c **hienptt22** để cấp tài khoản phù hợp cho dự án.
2. Dự án triển khai CI/CD cần đảm bảo điều kiện sau:
 - Có khả năng thực hiện build bằng lệnh. Ví dụ build bằng ant, maven, npm, msbuild, gradle,...
 - Dự án có server test để đảm bảo triển khai.
 - Dự án làm việc theo git flow để phù hợp cho việc tích hợp.
3. Các script cần chuẩn bị trước khi thực hiện cấu hình luồng triển khai trên jenkins.
 - Đối với CI/CD level 3 cần các script sau:

Tên	Ví dụ
Script build	<pre>mvn clean install npm install && npm run build ant build -f build.xml msbuild test.sln /T:Clean;Build /p:Configuration=Release /p:AutoParameterizationWebConfigConnectionString=False /p:DeployOnBuild=true" dotnet build docker build . -t \${habor}/etc/\${service}:\${version}</pre>

Script deploy ứng dụng	Sử dụng ansible. Cụ thể mô tả các bước deploy ứng dụng. Ví dụ: Copy file, stop/start ứng dụng. Tham khảo một số lệnh ansible tại link hướng dẫn ansible Run deploy ứng dụng với ansible: ansible-playbook cicd/deploy/deploy_ \${server}.yml -e groupId=\${groupId} -e artifactId=\${artifactId} -e BUILD_NUMBER=\${BUILD_NUMBER} Trong trường hợp có sử dụng môi trường k8s: ví dụ sử dụng lệnh kubectl -n \${namespace} apply -f \${service}-deployment* -- kubeconfig=\${configFile}
Script chạy unittest	mvn clean test org.jacoco:jacoco-maven-plugin:0.8.5:report-aggregate npm run coverage coverlet path/Tests.dll --target "dotnet" --targetargs "test -c Release --no-build --logger:"trx;LogFileName=TestResults.trx"" -f cobertura
Script automations test	Link git

III. Hướng dẫn chi tiết từng bước.

1. Bước 1: Tạo credential trên jenkins

CredentialID trong jenkins được sử dụng với một số mục đích sau:

- ✓ Authen để lấy source code gitlab. Dạng User/pass nhưng pass được sử dụng là Access token.
- ✓ Dùng để chứng thực đối với một số hệ thống như :
- ✓ Nexus: User/pass
- ✓ Harbor docker: user/pass
- a. Để tạo Credential thực hiện như sau:
 - Truy cập vào link sau để tạo: [đường dẫn](#) để tạo mới một credential: Chọn loại credential là user/pass
 - ✓ Nhập username đăng nhập gitlab
 - ✓ Nhập password là access token để clone source code.

Jenkins > Credentials > System > Global credentials (unrestricted)

Back to credential domains

Add Credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: hienptt22 Username đăng nhập gitlab

Password: Access token để clone source code

ID: hienptt22-gitlab ID được sử dụng trong các script để lấy dữ liệu trong gitlab hoặc jenkins

Description:

OK

- Trong trường hợp tạo credential cho account Nexus/Habor phần password sẽ nhập pass đăng nhập và đặt lại ID cho phù hợp.
- b. Trong trường hợp đã tạo credential từ trước và KHÔNG NHẬP ID của Credential thì thực hiện các thao tác sau để lấy credential id để sử dụng

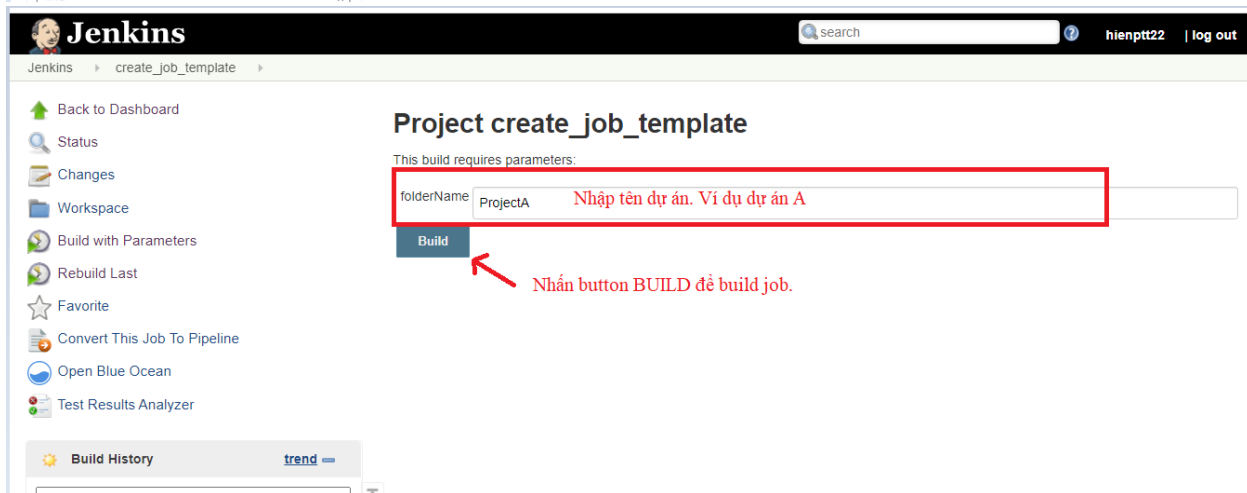
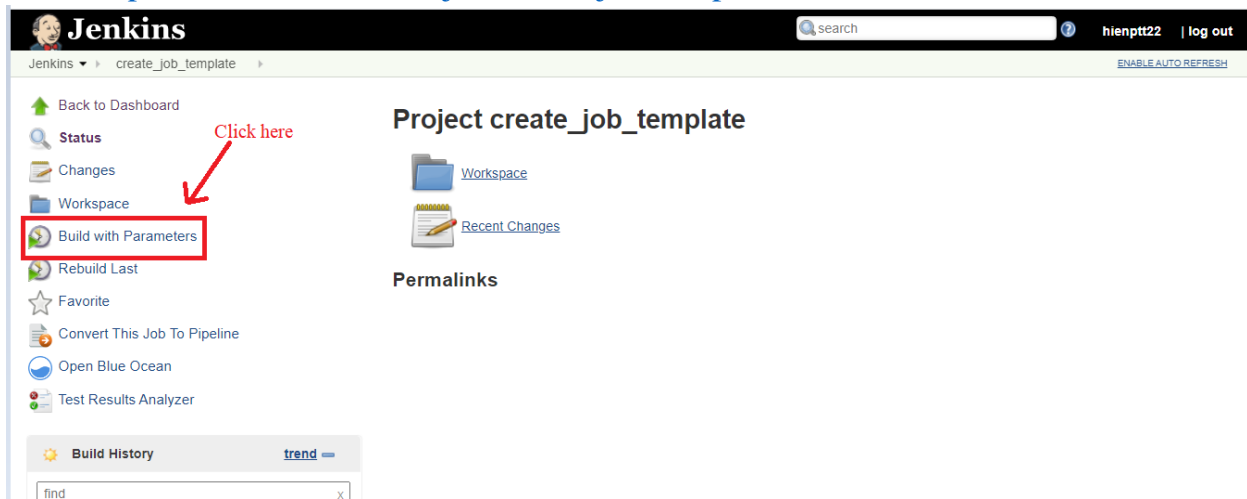


2. Bước 2: Tạo folder cho dự án trên jenkins.

Ở bước này P.KTPM sẽ hướng dẫn tạo 1 folder cho dự án và có sẵn các cấu hình mẫu. Thực hiện như sau:

- Đăng nhập và hệ thống jenkins và Truy cập job trên jenkins theo đường dẫn sau:

http://10.60.156.96:8080/job/create_job_template/



- Xem log build của job:

Jenkins > create_job_template > #1

Project create_job_template

[Workspace](#)
[Recent Changes](#)

Permalinks

- [Last build \(#1\) 24 sec ago](#)
- [Last stable build \(#1\) 24 sec ago](#)
- [Last successful build \(#1\) 24 sec ago](#)
- [Last completed build \(#1\) 24 sec ago](#)

Build History [trend](#)

find

#1 Mar 12, 2021 3:00 PM

[Changes](#) [RSS for failures](#)

Console Output

[View Build Information](#)

```
> Host: 10.60.156.96:8080
> Accept: */*
> Content-Type: text/xml
> Content-Length: 832
>
} [data not shown]
* upload completely sent off: 832 out of 832 bytes

100 832 0 0 100 832 0 829 0:00:01 0:00:01 --:--:-- 830
100 832 0 0 100 832 0 415 0:00:02 0:00:02 --:--:-- 415
100 832 0 0 100 832 0 276 0:00:03 0:00:03 --:--:-- 276
100 832 0 0 100 832 0 207 0:00:04 0:00:04 --:--:-- 207
100 832 0 0 100 832 0 166 0:00:05 0:00:05 --:--:-- 166
100 832 0 0 100 832 0 138 0:00:06 0:00:06 --:--:-- 0
100 832 0 0 100 832 0 118 0:00:07 0:00:07 --:--:-- 0
100 832 0 0 100 832 0 103 0:00:08 0:00:08 --:--:-- 0< HTTP/1.1 302 Found

< Date: Fri, 12 Mar 2021 08:00:54 GMT
< X-Content-Type-Options: nosniff
< Location: http://10.60.156.96:8080/job/ProjectA/configure
< Content-Length: 0
< Server: Jetty(9.4.z-SNAPSHOT)
<

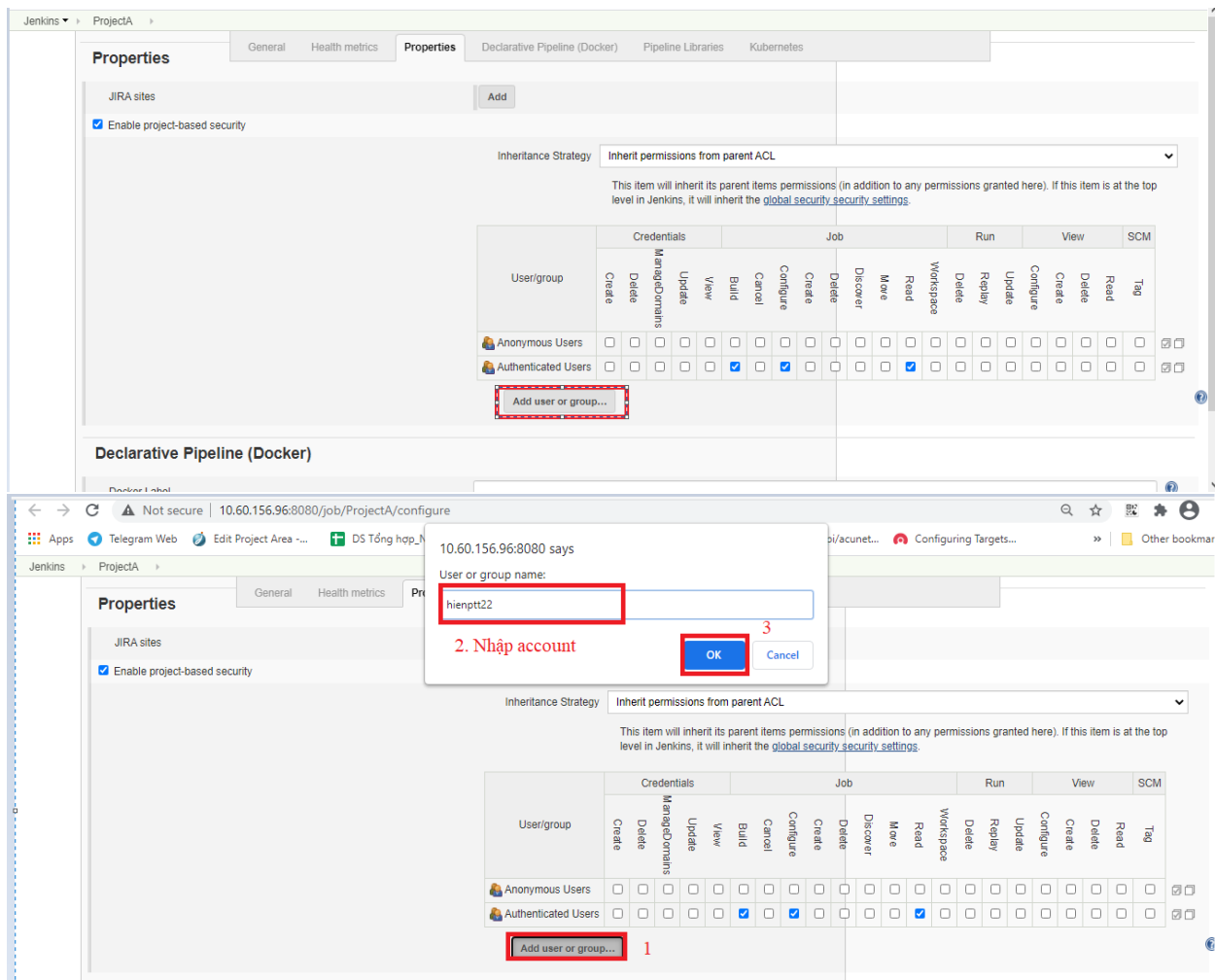
100 832 0 0 100 832 0 98 0:00:08 0:00:08 --:--:-- 0

* Connection #0 to host 10.60.156.96 left intact
[InfluxDB Plugin] Collecting data...
[InfluxDB Plugin] Publishing data to target 'InfluxDB' (url='http://10.60.156.104:8086', database='jenkins_db')
[InfluxDB Plugin] Completed.
Finished: SUCCESS
```

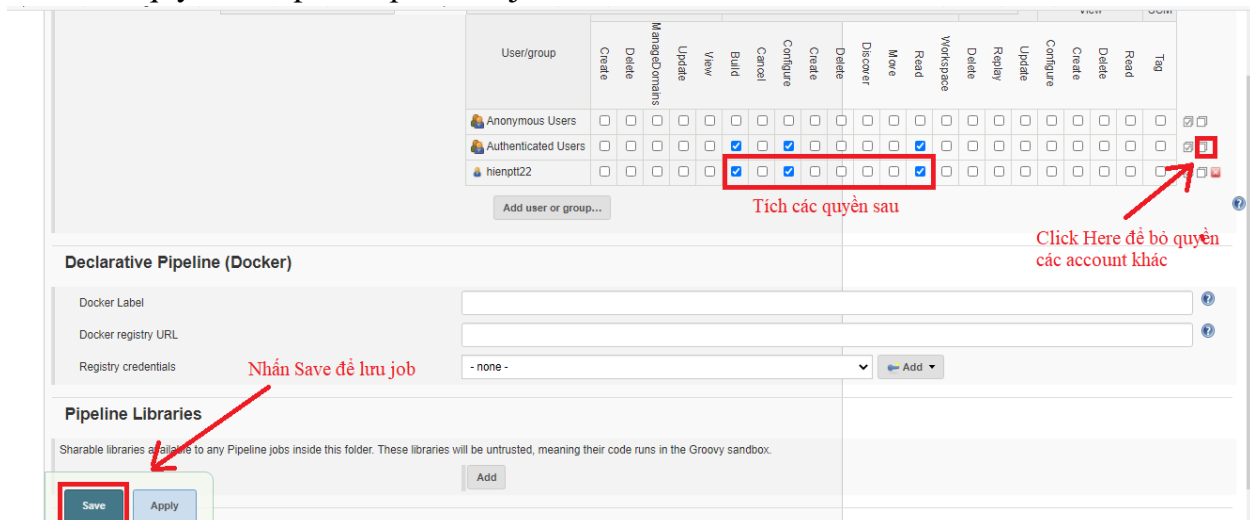
Click vào link job để thực hiện cấu hình

Lưu ý: Job sẽ chạy fail trong trường hợp folder đã tồn tại trên Jenkins. Trong trường hợp chạy fail thực hiện build lại và nhập lại tên project cho phù hợp.

- Thực hiện truy cập link job ở trên và thực hiện cấu hình folder như sau:
Sửa quyền view folder cho phù hợp với dự án. Chỉ nên để cho các thành viên của dự án có quyền view và cấu hình job. Quyền hiện tại đang để tất cả các account đã authen trên Jenkins đều có quyền build và cấu hình job.
Thực hiện add account vào job:



Cấu hình quyền cho phù hợp và lưu job.



Kết quả tạo folder cho dự án như hình:

Jenkins | ProjectA

Up, Status, Configure, New Item, People, Build History, Project Relationship, Check File Fingerprint, Convert Folders Job To Pipeline, Job Config History, Open Blue Ocean, Rename, Config Files, Credentials

ProjectA

All

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav	Number of builds
		DB	N/A	N/A	N/A		
		service_example	N/A	N/A	N/A		

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

3. Bước 3: Thực hiện cấu hình luồng CI/CD cho một service.

- Thực hiện đổi tên folder **service_example** ➔ thành tên service của dự án như sau:

Jenkins | ProjectA | service_example

Up, Status, Configure, New Item, People, Build History, Project Relationship, Check File Fingerprint, Convert Folders Job To Pipeline, Job Config History, Open Blue Ocean, **Rename**, Config Files, Credentials

service_example

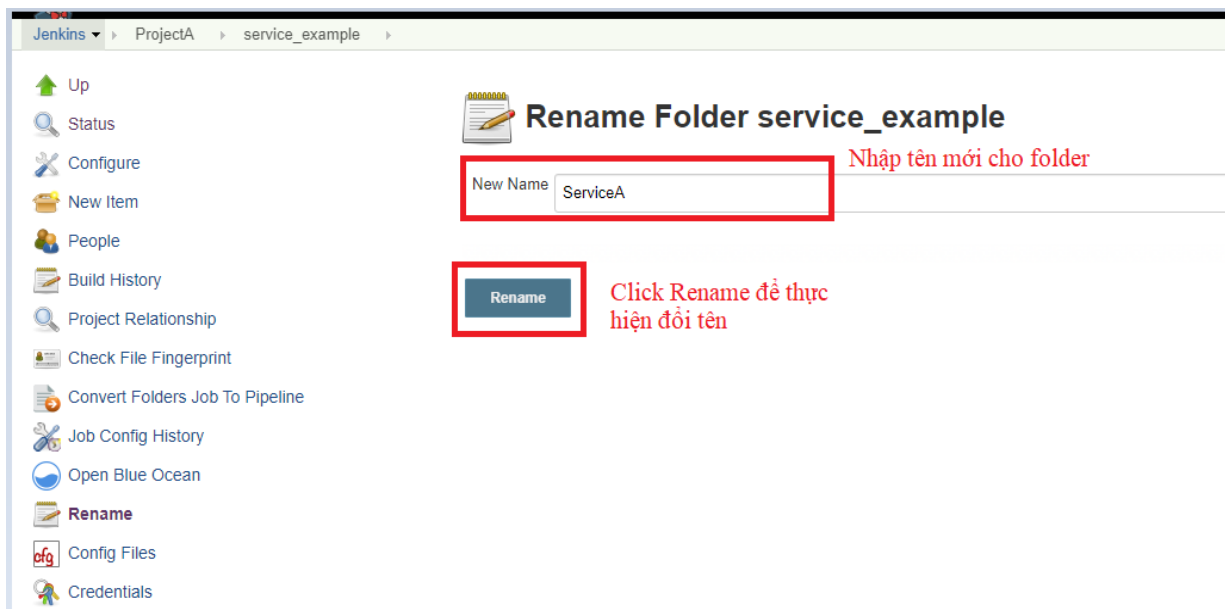
All

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav	Number of builds
		CD	N/A	N/A	N/A		0 0 0
		CD_Pre	N/A	N/A	N/A		0 0 0
		CI_dev	N/A	N/A	N/A		0 0 0
		CI_staging	N/A	N/A	N/A		0 0 0

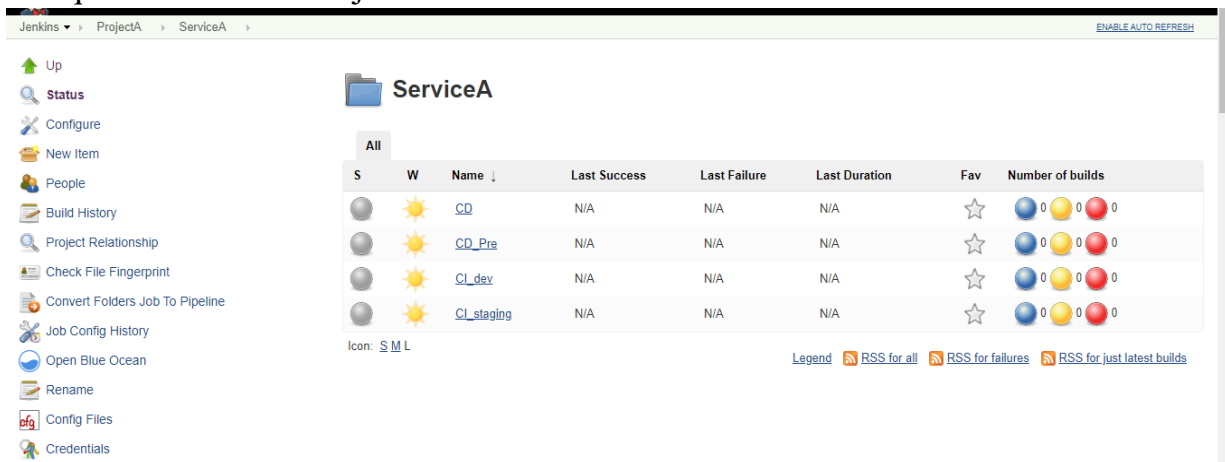
Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

Click Rename job



Kết quả sau khi Rename job như sau:



4. Thực hiện cấu hình job.

Cấu trúc thư mục của 1 service sẽ bao gồm các job và chức năng cụ thể như sau:

- CI_dev: job chạy khi có sự kiện push/merge request tới nhánh Dev. Luồng này sẽ được cấu hình để build và deploy ứng dụng lên server test.
- CI_staging: job chạy khi có sự kiện push/merge request tới nhánh master. Luồng này sẽ được cấu hình để build và deploy ứng dụng lên môi trường staging
- CD: job chạy khi có sự kiện TAG trên git. Luồng này sẽ được cấu hình để deploy ứng dụng lên môi trường productions.

Để có thể chạy được job. Mỗi folder sẽ có 1 file config để lưu các biến môi trường. Đối với các dự án cần sửa lại cho phù hợp các tham số như sau:

- Thực hiện cấu hình và sửa configFile:

Jenkins > ProjectA > ServiceA > Config Files

ENABLE AUTO REFRESH

CI_dev: job chạy trên jenkins khi có sự kiện push/merge request tới nhánh dev
 CI_staging: job tích hợp khi có sự kiện push/merge request tới nhánh master
 CD: job deploy lên productions

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav	Number of builds
		CD	N/A	N/A	N/A	☆	0 0 0
		CD_Pre	N/A	N/A	N/A	☆	0 0 0
		CI_dev	N/A	N/A	N/A	☆	0 0 0
		CI_staging	N/A	N/A	N/A	☆	0 0 0

Icon: S M L

Legend RSS for all! RSS for failures RSS for just latest builds

Jenkins > ProjectA > ServiceA > Config Files

Manage Jenkins
 Config Files
 Add a new Config

Config File Management

You can edit or remove your configuration files

Groovy file

ci-config

Thực hiện sửa các tham số sau cho phù hợp với từng dự án

env.groupName="DEMO_CICD" //nhập tên service đối với dự án nhiều service. Còn nếu dự án 1 module nhập tên dự án

env.node_slave="slave_116" //nhập tên node chạy job (slave_43/slave_116/node_cicd). Đối với dự án MsBuild chạy trên node (slave_203)

env.GITLAB_PROJECT_API_URL="http://10.60.156.11/api/v4/projects/[Nhập project Id của dự án trên gitlab]"

env.ROLLBACK_MAINTAINER_LIST = '[Nhập maintainer dự án]'

env.project_maintainer_list = '[Nhập maintainer dự án]'

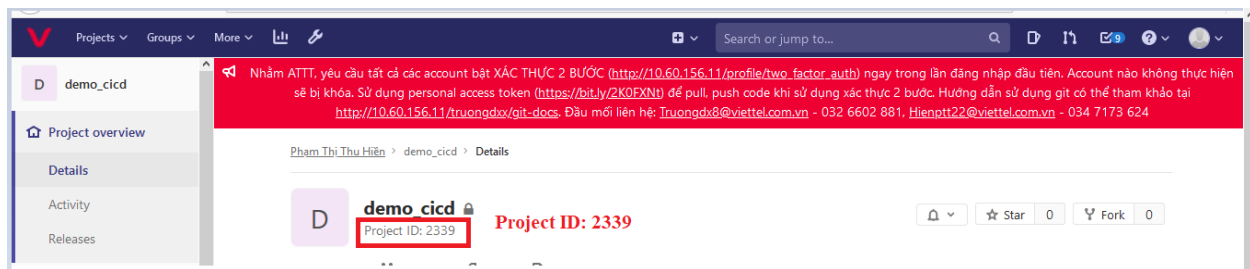
env.buildUrlCI='[Link job staging trên jenkins]/lastBuild' //link sử dụng cho job CD

env.buildUrlStaging='[Nhập maintainer dự án]' //link sử dụng cho job CD

Hướng dẫn lấy projectID của gitlab để điền vào phần

env.GITLAB_PROJECT_API_URL như sau:

Truy cập link gitlab của dự án:

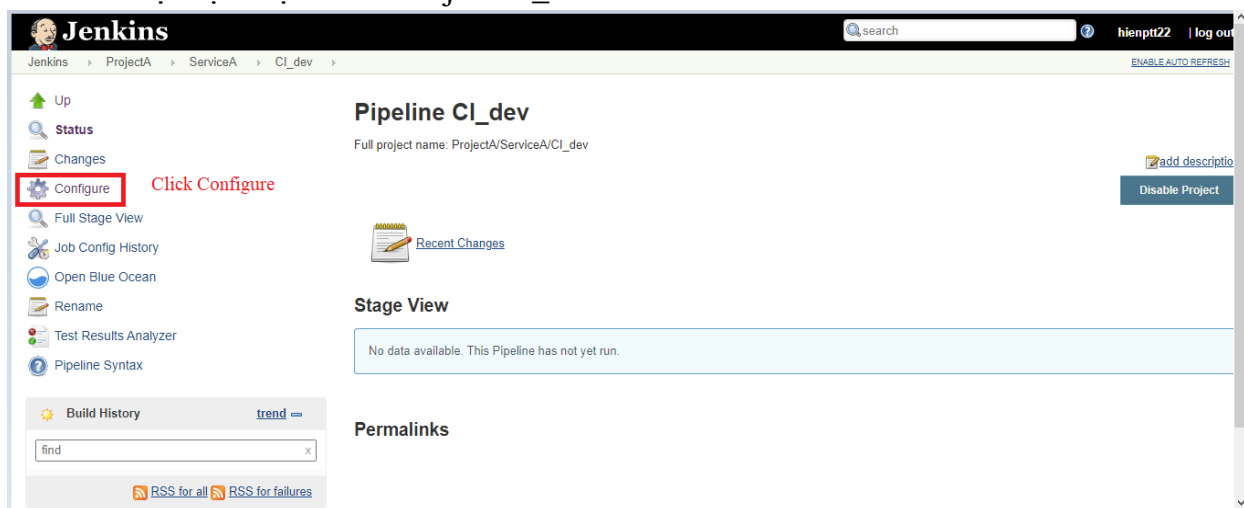


Thực hiện sửa giá trị tương ứng với giá trị lấy được như sau:

env.GITLAB_PROJECT_API_URL="http://10.60.156.11/api/v4/projects/2339"

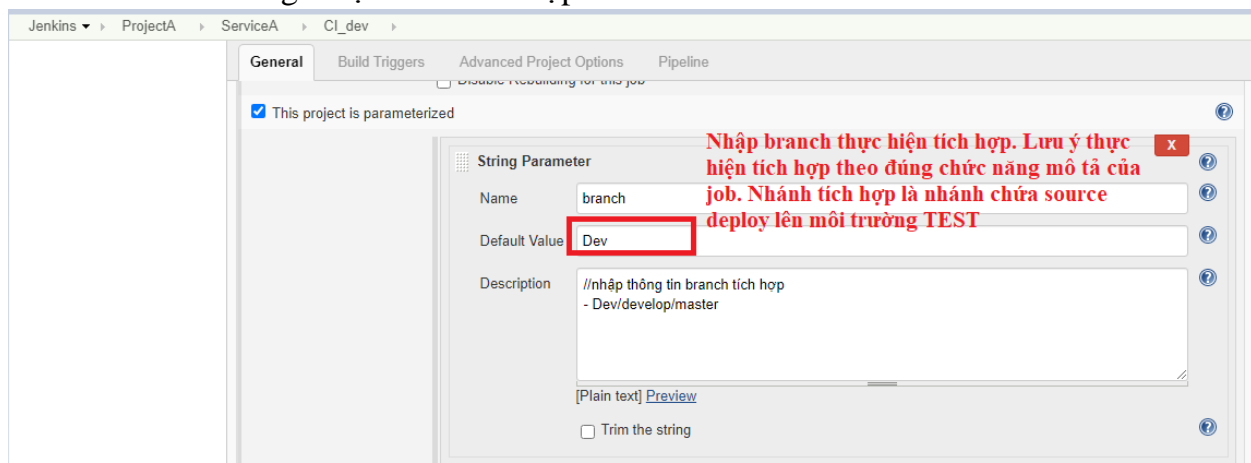
5. Thực hiện cấu hình job CI/CD trên Jenkins:

Ví dụ thực hiện cấu hình job CI_dev:



Thực hiện sửa một số tham số sau cho phù hợp với dự án:

- Sửa giá trị nhánh tích hợp:



- Sửa slave để chạy job. Sửa tương ứng với giá trị “env.node_slave” cấu hình trong configFile.

Jenkins ▾ ▸ ProjectA ▸ ServiceA ▸ CI_dev ▸

General Build Triggers Advanced Project Options Pipeline

☐ Trim the string

String Parameter X ?

Name Giá trị tương ứng với giá trị biến env.node_slave trong configFile

Default Value

Description
[Plain text] [Preview](#)

☐ Trim the string

Add Parameter ▾

- Cấu hình lựa chọn các Gitlab event mà job sẽ xử lý

☐ Build after other projects are built

☐ Build periodically

☒ Build when a change is pushed to GitLab. GitLab webhook URL URL được sử dụng để cấu hình trong gitlab webhook

Enabled GitLab triggers

Push Events ☒

Opened Merge Request Events ☒ Các event được cấu hình để chạy job

Accepted Merge Request Events ☐

Closed Merge Request Events ☐

Rebuild open Merge Requests ▾

Approved Merge Requests (EE-only) ☒

Comments ☒ Comment recheck trong merge request để thực hiện rebuild

Comment (regex) for triggering a build

Chọn advanced để cấu hình các tham số khác Advanced...

General **Build Triggers** Advanced Project Options Pipeline

Build on successful pipeline events ☐

Pending build name for pipeline

Cancel pending merge request builds on update ☐

Allowed branches

☐ Allow all branches to trigger this job

☒ Filter branches by name **Nhánh tích hợp**

Include **develop** **ERROR**

Exclude

☐ Filter branches by regex

☐ Filter merge request by label

Secret token **Secret để cấu hình gitlab webhook**

0c12b9a8ad3bb50afe63c8439af1341a

Generate

Click generate để sinh lại token

Clear

Lưu ý: Nội dung pipeline script không được sửa. Nếu sửa sẽ cần approve của Admin.

Pipeline

Definition Pipeline script

Script

```

1 node("$node"){
2     checkout changelog: true, poll: true, scm: [
3         $class      : 'GitSCM',
4         branches     : [[name: "$branch"]],
5         doGenerateSubmoduleConfigurations: false,
6         extensions    : [[class: 'UserIdentity', email: 'ci@jenkins.io']],
7         submoduleCfg  : [],
8         userRemoteConfigs : [[credentialsId: 'a5eadd9f-332d-4575-8000-000000000000',
9                               name      : 'origin',
10                              url       : "${env.gitlabSource}"]]
11     ]
12     jenkinsfile_bootstrap = load 'jenkinsfile_bootstrap.groovy'
13     jenkinsfile_bootstrap.bootstrap_build()
14 }

```

☐ Use Groovy Sandbox

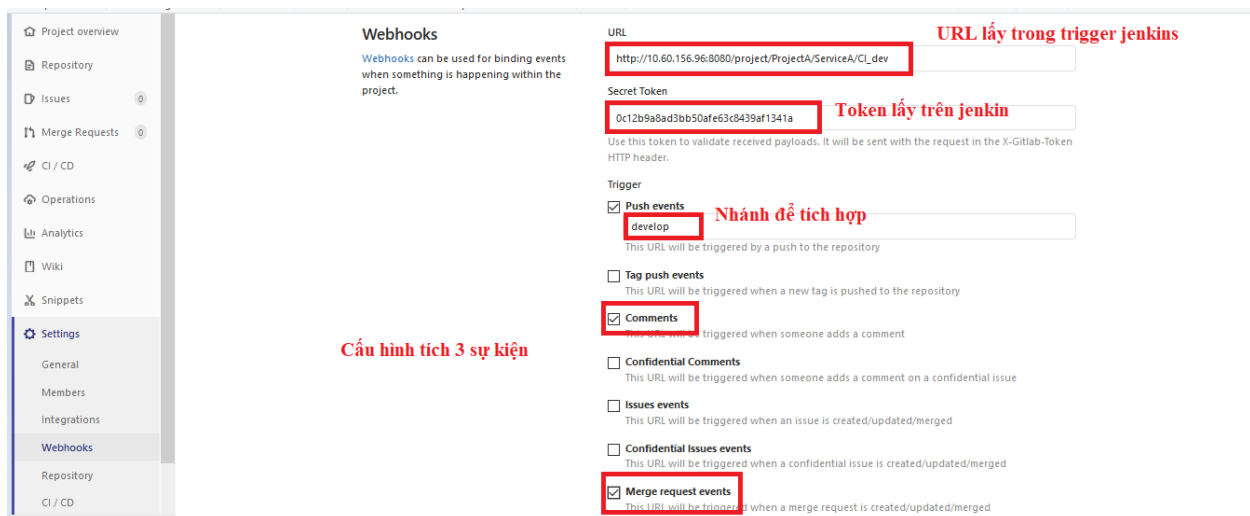
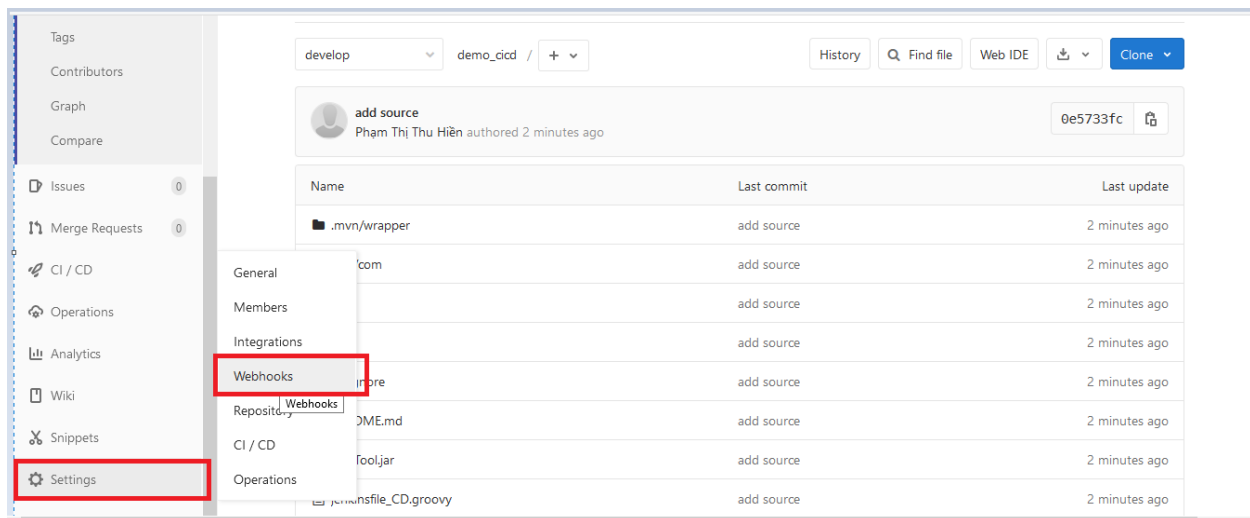
[Pipeline Syntax](#)

6. Cấu hình trigger gitlab với jenkins

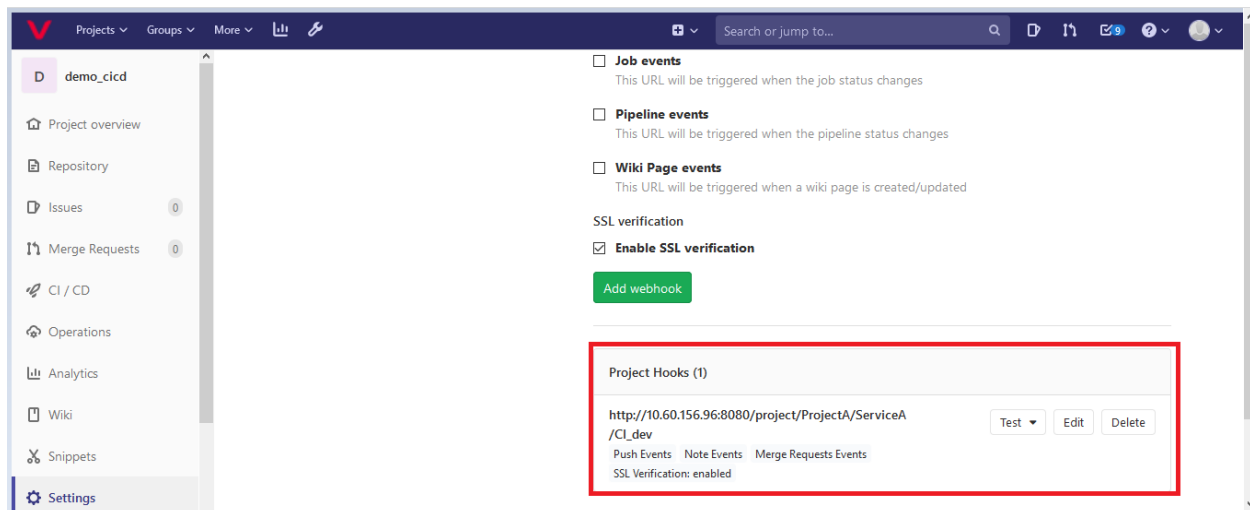
Lưu ý: Webhook của gitlab chỉ có maintainer mới có quyền cấu hình.

Thực hiện cấu hình như sau:

- Trên Gitlab, vào **Settings** → **Webhook Settings**



Webhook sau khi cấu hình:








7. Lưu job jenkins:

8. Thực hiện viết jenkinsfile để chạy job

- Lấy các 4 file jenkinsfile trên gitlab tại đường dẫn sau:

http://10.60.156.11/hienptt22/TaiLieuAuto/-/tree/master/Module%20Config%20pipeline%2Fresource%2Fci-cd-script-example%2Fjenkinsfile_groovy_update

 Jenkinsfile_init	cicd level 3,4	2 weeks ago
 jenkinsfile_CD.groovy	cicd level 3,4	2 weeks ago
 jenkinsfile_CI.groovy	cicd level 3,4	2 weeks ago
 jenkinsfile_bootstrap.groovy	cicd level 3,4	2 weeks ago
 jenkinsfile_utils.groovy	cicd level 3,4	2 weeks ago

Có các file sau:

- **jenkinsfile_bootstrap.groovy**: định nghĩa các sự kiện chạy job, configFile
- **jenkinsfile_CD.groovy**: định nghĩa các bước cho deploy lên productions
- **jenkinsfile_CI.groovy**: định nghĩa các stage chạy cho sự kiện push/merge request.
- **jenkinsfile_utils.groovy**: các hàm dùng chung.

Tổ chức thư mục chứa source bao gồm các file và folder sau:

- Source code
- 4 file **jenkinsfile**
- Folder tên **cicd**

Ví dụ như sau:

Name	Date modified	Type	Size
.mvn	3/12/2021 3:51 PM	File folder	
cicd	3/12/2021 3:51 PM	File folder	
Libs	3/12/2021 3:51 PM	File folder	
src	3/12/2021 3:51 PM	File folder	
.gitignore	3/12/2021 3:51 PM	Text Document	2 KB
jenkinsfile_bootstrap.groovy	3/12/2021 3:51 PM	GROOVY File	32 KB
jenkinsfile_CD.groovy	3/12/2021 3:51 PM	GROOVY File	0 KB
jenkinsfile_CI.groovy	3/12/2021 3:51 PM	GROOVY File	18 KB
jenkinsfile_utils.groovy	3/12/2021 3:51 PM	GROOVY File	6 KB
mvnw	3/12/2021 3:51 PM	File	10 KB
mvnw.cmd	3/12/2021 3:51 PM	Windows Comma...	7 KB
nbactions.xml	3/12/2021 3:51 PM	XML Document	2 KB
pom.xml	3/12/2021 3:51 PM	XML Document	11 KB
README.md	3/12/2021 3:51 PM	MD File	13 KB
settings.xml	3/12/2021 3:51 PM	XML Document	2 KB
SmsTool.jar	3/12/2021 3:51 PM	Executable Jar File	1,922 KB

Tất cả các script build và deploy sẽ được gọi trong file **jenkinsfile_CI.groovy**

- **Build**

```
def buildService(buildType, gitBranch) {
    stage("Checkout Source Code") {
        jenkinsfile_utils.checkoutSourceCode(buildType)
        echo 'Checkout source code'
    }
    stage('Build jar file & Build Docker'){
        def folder = sh(script: 'pwd', returnStdout: true)
        env.buildFolderResult = folder.trim()
        def pomVersion = readMavenPom([file: "pom.xml"]).getVersion()
        env.versionProject =
"${gitBranch}_${pomVersion}_u${BUILD_NUMBER}"
        // sh ./cicd/build.sh ${env.versionProject}
        sh """
            mvn clean install -DskipTests //sửa lệnh build cho phù hợp.
            """
    }
}
```

- Push bản build tới Nexus

```

def packageServicesAndUploadToRepo(groupId, artifactId, moduleName){

  stage('Upload artifact to Nexus server'){
    sh 'pwd'
    def uploadSuccessComment = "<b>Build & package Artifact Results - " +
      "Build Artifact module ${moduleName} is created. "
    nexusArtifactUploader artifacts: [[artifactId: "${artifactId}_${moduleName}",
classifier: "", file: "target/serviceTelecare-0.0.1-SNAPSHOT.jar", type: 'jar']],
credentialsId: "${env.NEXUS_CREDENTIALID}", groupId: "${groupId}", nexusUrl:
'10.60.156.26:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'msbuild',
version: "1.${BUILD_NUMBER}"
    env.PACKAGE_UPLOAD_IMAGE_RESULT_STR = uploadSuccessComment
  }
}

```

Lưu ý: sửa đường dẫn file kết quả build và loại file cho phù hợp

- Deploy: lệnh để thực hiện deploy ứng dụng

```

def deploy_module_web(server,groupId,artifactId){

  echo "deploy to server ${server}"

  sh """

    pwd

    ansible-playbook cicd/deploy/deploy_${server}.yml -e groupId=${groupId} -
e artifactId=${artifactId} -e BUILD_NUMBER=${BUILD_NUMBER}

  """

}

```

9. Định nghĩa các stage cho từng sự kiện. Ví dụ như sau

```

def buildPushCommit() {

  echo "gitlabBranch: ${env.gitlabBranch}"

  def tasks = [:]

  tasks['unitTestAndCodeCoverage'] = {

    node("${env.node_slave}") {

      echo "Unit Test"
    }
  }
}

```

```

        // unitTestAndCodeCoverage("PUSH")
    }
}
tasks['SonarQubeScan'] = {
    node("$env.node_slave") {
        sonarQubeScan("PUSH")
    }
}
tasks['Package and Build Artifact'] = {
    node("$env.node_slave") {
        buildService("PUSH", "$env.gitlabBranch")
    }
}
parallel tasks
if(env.gitlabBranch == env.STAGING_BRANCH){
    dir("$env.buildFolderResult"){
        stage("Push Artifact To Nexus"){
            node("$env.node_slave"){

packageServicesAndUploadToRepo("DEMO_CICD","CI_staging","Back-End")
            }
        }
        stage("Deploy to Server"){
            node("$env.node_slave"){
                deploy_module_web("staging","DEMO_CICD","CI_staging_Back-End")
            }
        }
    }
}

```



```

    }
} else {
    dir("$env.buildFolderResult"){
        stage("Push Artifact To Nexus"){
            node("$env.node_slave"){

packageServicesAndUploadToRepo("DEMO_CICD","CI_staging","Back-End")
            }
        }
        stage("Deploy to Server"){
            node("$env.node_slave"){
                deploy_module_web("dev","DEMO_CICD","CI_staging_Back-End")
            }
        }
    }

}

currentBuild.result = "SUCCESS"

}

```

10. Push commit và chạy thử luồng